Global Information Assurance Certification Paper

GIAC Reverse Engineering Malware
GREM Practical Assignment Version 1.0

Reverse Engineering MSRLL.EXE

Bryan Fendley

December 2004

Table of Contents

Abstract

This paper is the practical assignment necessary for the completion of the SANS ILOT Certificate in Reverse Engineering Malware. The paper will demonstrate the process of methodically analyzing a malware specimen in a controlled environment. The paper will demonstrate the methodology explained during the course as well as demonstrate an example testing environment and softwares that can be used during the analysis process. Many of the programs used are free. Some software is not free but trial versions are available.

Section 1: Laboratory Setup

The following is a description of the laboratory setup including networking, hardware, and software that I used during the course and for the analysis of the practical assignment in reverse engineering malware.

Section 1.1 Operating Systems

Operating systems used during my analysis included Microsoft Windows XP and Red Hat Linux 2.4.20-8. The Linux operating system was used to run an IRC server and Telnet during this particular analysis to attempt to communicate and issue commands to the malware specimen. Windows XP was used as a host environment for the malware. Windows XP was also used as the operating system to host the VMWare virtual machines.

Section 1.2 Network Configuration

In order to insure safe handling of the malware specimen the network configuration for my analysis was self contained. VMWare was used to create and host virtual machines on a single computer, and to establish a virtual network in a dual homed host only environment in which VMWare provided DHCP services.

I used VMWare workstation version 4.5.1 to run the following operating systems as guest operating systems in the emulated VMWare host only network.

VMWare's host only networking option helped in providing an isolated network. When VMWare is configured in host only mode the Virtual Operating Systems function as if on an isolated hub based network.

VMWare provided DHCP services to the virtual operating systems with IP addresses assigned as follows:

Virtual machine 1 was a Microsoft Windows XP version 2002 un-patched, IP address: 192.168.159.130. Virtual machine 2 was a Linux Red Hat image provided as part of the course, IP address: 192.168.159.137. Figure 1-1 illustrates the virtual network configuration used during the analysis of msrll.exe.

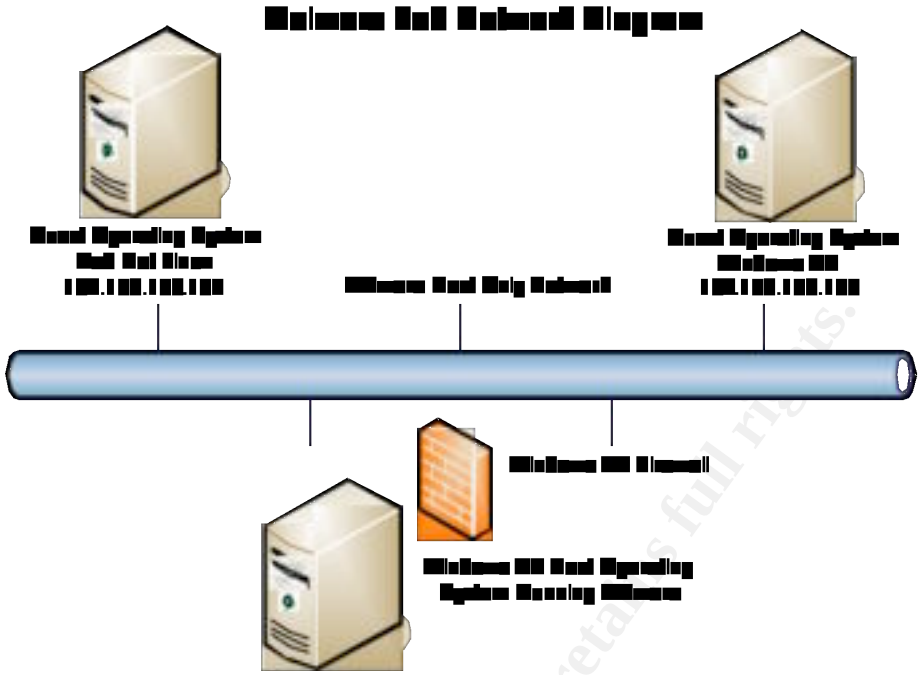

Figure1-1 Network Diagram

VMWare was also very important because it gave me the ability to backup and restore full systems using the “snapshot” and “revert to snapshot features”. This feature allowed me to execute and observe the malware repeatedly in a controlled environment.

Section 1.3 Physical Configuration

For the physical configuration of my analysis environment, I used two PCs connected to a single monitor and keyboard using a kvm switch. This configuration allowed me to switch easily between my test environment and production environment, and helped to conserve space in my office.

The first PC is my regular workstation and is connected to the internet. I used it for research, tool downloading, report writing etc. during the analysis.

The second PC used for analysis was a fully patched Intel Pentium III, 927 MHz processor with 512 MB RAM, running Microsoft Windows XP version 2002 with the firewall enabled to further protect the host OS from any malware that would be examined on the virtual OS and network. In order to make sure no malware is able to escape into the wild, it is also very important that this system does not have access to your production network or the internet in any way.

Both PCs and virtual OS have access to a printer via a parallel 4 port printer switch. I found it useful during analysis to be able to print from my production system and analysis system. In order to transfer necessary files to my analysis machine, I used a CD and thumb drive. Figure 1-2 illustrates the physical configuration of my malware analysis lab.

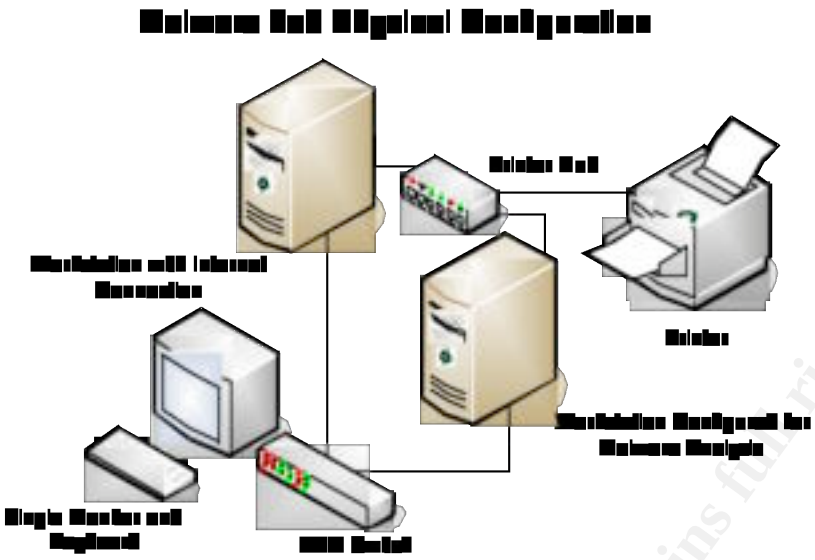

Figure1-2

Section 1.4 Three Phase Analysis Model
During my analysis I used a three phase model to attempt to define characteristics of the malware specimen and to identify and control it's behaviors in this order: property analysis, behavioral analysis, and code analysis.

I configured my virtual Operating System that was to be infected with the following list of tools and then saved a "snap shot" using VMWare. Since I was in the process of learning to do malware analysis, I used the "revert to snapshot" feature extensively in order to get back to a baseline. This particular process was extremely important and useful during my process of learning to use the tools for reverse engineering malware.

Section 1.5 Tools Used for Property Analysis
For the property analysis phase I used the command line utility MD5sum to uniquely identify the malware specimen by computing its MD5 hash. I also used Windows properties viewer to identify properties of the malware specimen such as size and creation date.

Section 1.6 Tools Used for Behavioral Analysis
In order to analyze the behavioral effects of the malware on the file system and registry, I used a program called InstallWatch Pro to create a log of changes made to the file system and registry during the installation of the malware specimen. I also used a program called FileMon to monitor file changes during the installation of the malware specimen. RegMon was another program package I used to monitor registry changes during the installation of the malware specimen. I also used a program called RegShot to compare registry settings before and after the installation of the malware specimen. Also used was AutoRuns a tool that will allow you to see processes set to start at windows start

up. AutoRuns will also allow you to easily navigate to the processes registry setting for editing. The use of all of these tools may be unnecessary since some may provide similar results. Since I was in learning mode I wanted to try different tools. I also wanted to be able to compare results of similar tools for accuracy.

In order to analyze the network behaviors of the malware specimen, I used the command line utility NetStat with the "-a" switch to list port status on the infected system before and after the execution of the malware specimen. I also used a program called TDIMon to monitor port changes during the installation of the malware specimen. I used the Ethereal packet sniffing utility on my host OS to capture and analyze any packets sent by the malware specimen.

In order to analyze the malware specimen's behavior further I used Telnet a common terminal emulation program to try to communicate with the malware specimen. During my analysis I used Telnet to attempt to issue commands to the malware specimen. Since it is common for malware to use Internet Relay Chat channels to communicate, an IRC server (Internet Relay Chat server) was also used on my virtual Linux machine to try to communicate and issue commands to the malware specimen.

Section 1.7 Tools Used for Code Analysis

During Code analysis I used AspackDie to unpack the malware specimen. AspackDie is an unpacker for PE files which were compressed using any version of Aspack since Aspack 2000. Aspack is a win32 executable file compressor utility used for reducing file size and protecting against reverse engineering.

OllyDbg is a 32-bit assembler level analyzing debugger that runs on Microsoft Windows. I used it to examine the malware specimen's code in order to attempt bypassing the malware specimen's authentication and to attempt to issue commands to msrll.exe.

BinText was the program I used to extract human readable strings from the malware specimen in its packed and unpacked versions. These strings would provide me with action words and clues with their associated memory addresses that could be used during the code analysis phase.

A program called LordPE was also used. LordPE is a tool designed to assist in the viewing of PE files dumped from memory allowing them be analyzed and edited.

Section 2: Properties of the Malware Specimen

I downloaded the file from the SANS website and transferred it to my analysis system via thumb drive. I unzipped the file using WinZip and placed it at the root of C on my virtual Windows XP system.

The name of the file was msrll.exe. To find out more details, I right clicked on the file and obtained the following details: Windows Application File, Size 41.0 kb, created May 10, 2004, at 4:29:54 p.m. indicating that the file was designed to run on a Windows operating system.

My next step was to obtain the MD5sum of the Msrll.exe file using the command line tool MD5sum.exe. As seen in figure 1-3 the MD5sum was equal to: 84acfe96a98590813413122c12c11aaa.

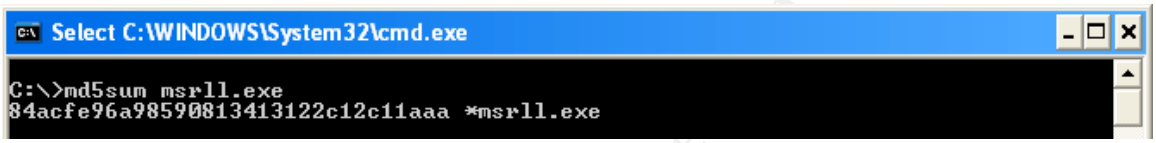

Figure 1-3 Md5sum of msrll.exe

This provided me with a unique hash that would allow me to monitor if the file changes in any way during execution.

I then used BinText to examine the malware executable for readable text strings. To do this I opened the BinText program and then under “file to scan” I browsed to C:\msrll.exe and pressed the “GO” button. This produced a report of readable strings. I found one useful text string. As seen in figure 1-4, I could see that one string was “.aspack”. This told me that the executable was compressed using Aspack compression. I would later unpack this file and run BinText again to see more useful strings.

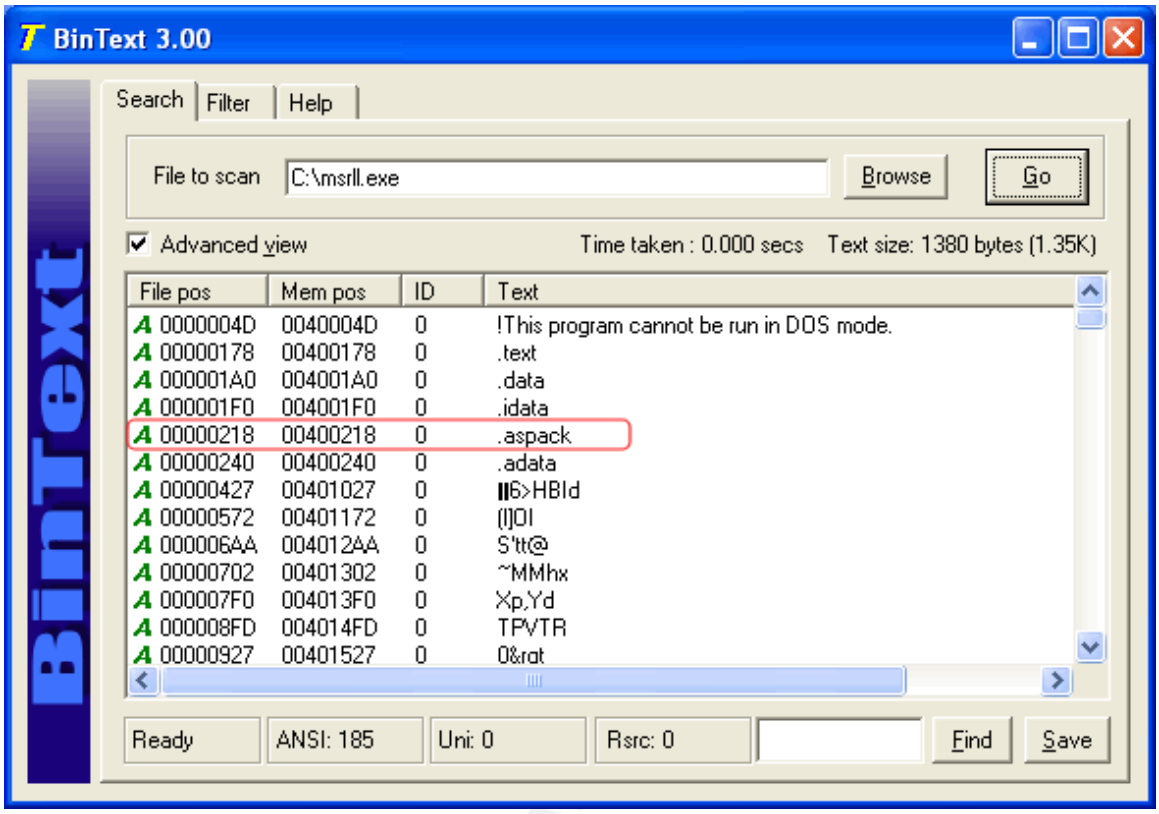

Figure1-4 BinText Results

I then opened the msrll.exe file in Ollydbg and used the Ctrl N function to list all symbolic names. I again saw the string “aspack” as seen in figure 1-5, further verifying that msrll.exe was compressed using Aspack.

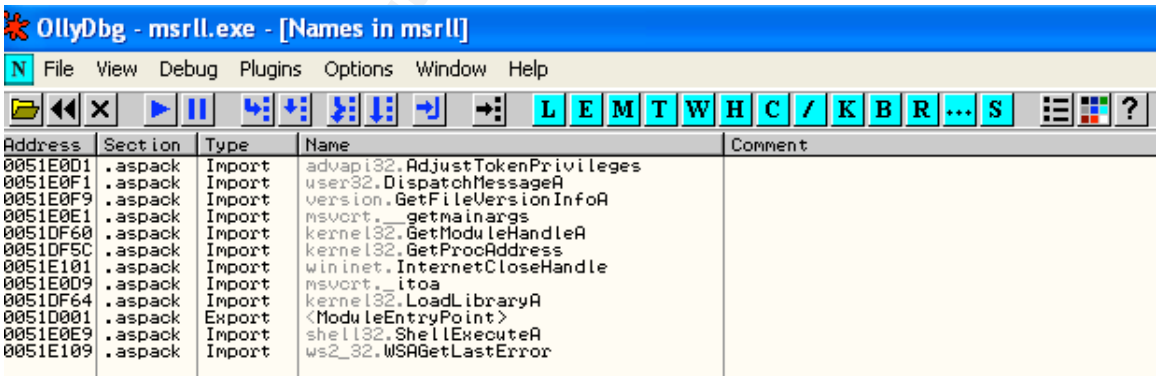

Figure1-5 Symbolic Names with OllyDbg

Note: in real world analysis, you would also want to consult anti-virus websites regarding the properties of a particular malware specimen. Since I was looking to test my new skills, I purposefully skipped this process during this particular phase of my analysis.

After viewing and recording the initial properties of the malware specimen, it was time to move on to the behavioral analysis phase.

Section 3: Behavioral Analysis

Using a program package called InstallWatch Pro 2.5, I installed msrll.exe by pressing InstallWatch Pro's install button and browsing to the msrll.exe file that was located at the root of C on my Windows XP virtual system. InstallWatch records changes made to your PC during program installation. According to InstallWatch, 5 files were added, 2 files deleted, 4 files updated, 27 registry entries added, 0 registry entries deleted, and 16 registry entries updated. Note: By using the snap shot feature within VMWare I was able to try this process more than once and received slightly varying results as far as the number of files and registry entries created, deleted, and modified. Figure 3-1 shows the results of one attempt to capture changes using InstallWatch.

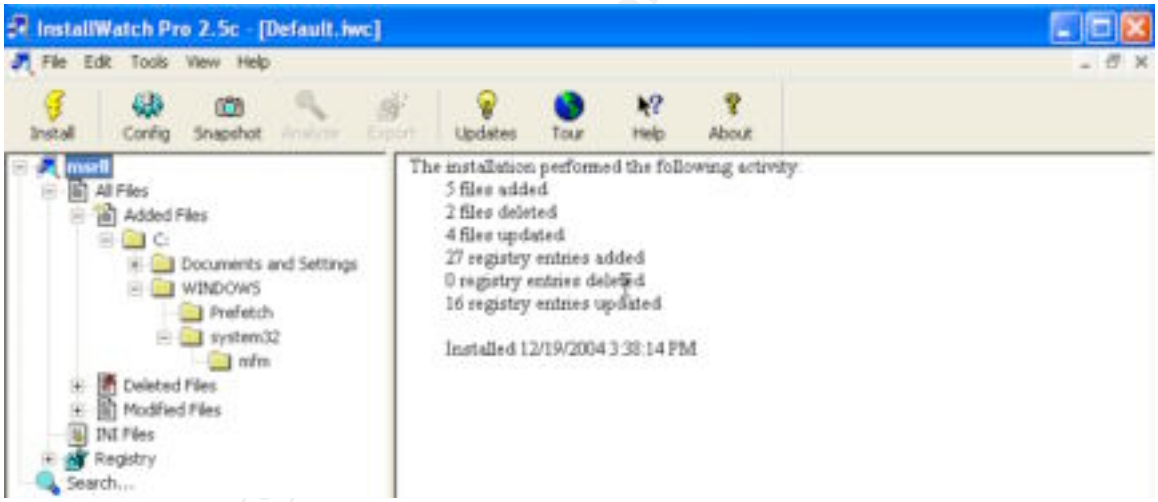

Figure3-1

By using InstallWatch I was able to see a folder named MFM 1 kb in size was created in the System 32 folder. The MFM folder contained 2 files: jtram.conf 2kb created on 11/19/2004 (the date of the analysis) and msrll.exe 42kb created 5/10/2004 (a date prior to my analysis). Figure 3-2 shows the InstallWatch report of added files.

The file jtram.conf was located in the "mfm" folder along with the msrll.exe file. I opened it with Windows notepad, but could not make out what it was. There were lots of numbers and letters in indistinguishable patterns. I checked the properties of this file at later times during the analysis and it appeared that it was being modified.

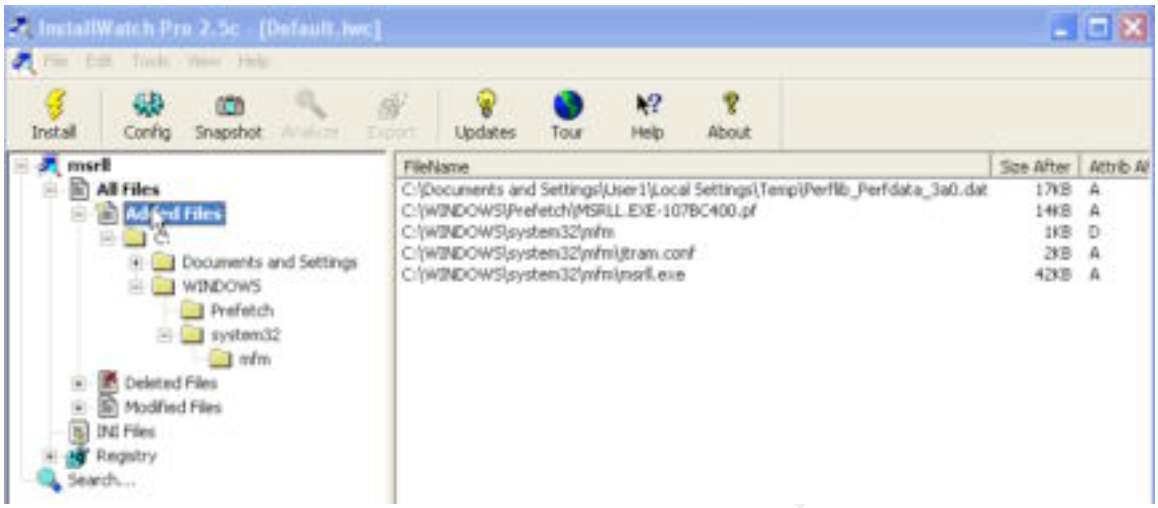

Figure3-2

To see if the file was modified in anyway I obtained the MD5sum of the newly created msrll.exe file using the command line tool MD5sum.exe. The MD5sum matched the original msrll.exe file: 84acfe96a98590813413122c12c11aaa.

After the msrll.exe was executed, I noticed that it had removed itself from its original location at the root of C as seen in figure 3-3. This was verified with InstallWatch and later with Filemon as seen in figure 3-4.

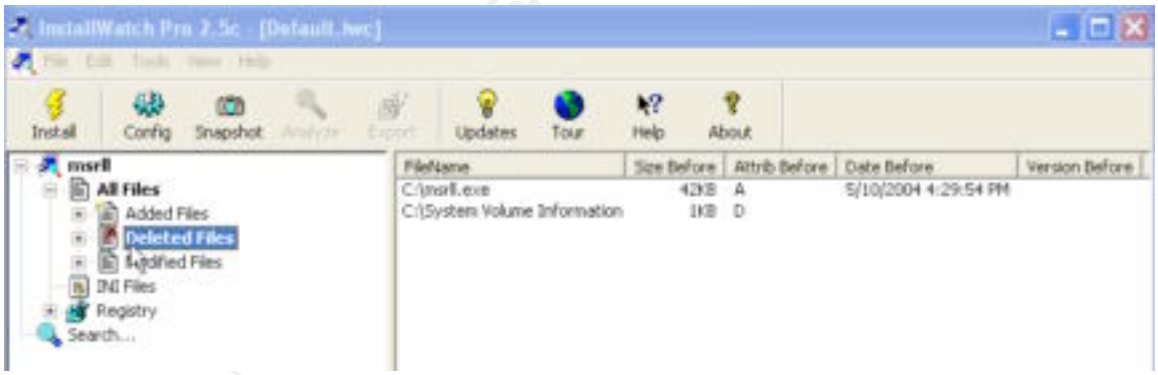

Figure3-3

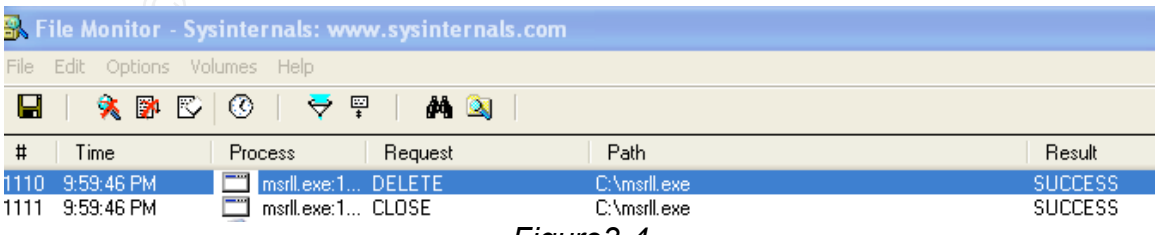

Figure3-4

Using AutoRuns to show all services set to automatically start, I noticed there was now a listing for msrll.exe as seen in figure 3-5.

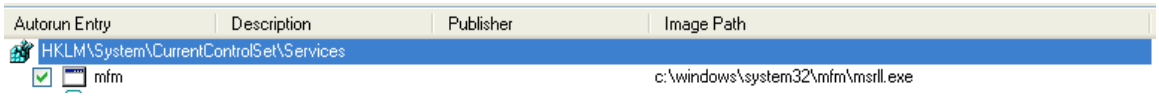

Figure3-5

While in AutoRuns I clicked the service listing for “mfm” as seen in figure 3-5. I was able to easily get to the registry editor and view detailed listings for the newly created service as seen in figure 3-6.

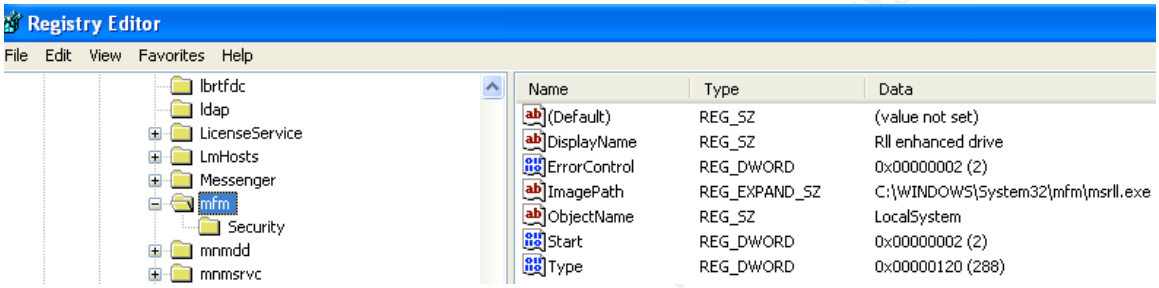

Figure 3-6

As seen in figure 3-7, I now had a registry setting for msrll.exe to start as a service: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm with an image path pointed to C:\WINDOWS\System32\mfm\msrll.exe with a display name of “Rll enhanced drive”

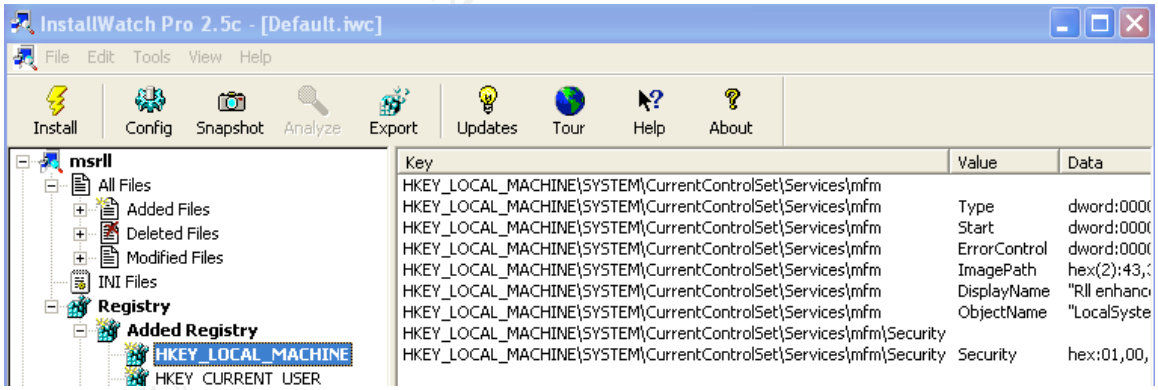

Figure3-7

Next I used the following tools to confirm the results of InstallWatch Pro and AutoRuns.

I used the “revert to snap shot feature” in VMWare to get back to a known good state.

I then used Regshot to take a picture of my registry before and after the install of msrll.exe. Then using the compare feature of Regshot, I see that Regshot confirmed the findings of AutoRuns in regards to the registry setting for: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm. RegShot was also helpful in providing results for files added and deleted as seen in figure 3-8.

```
Reg shot

---------------------------------
Keys added:7
---------------------------------
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Services\mfm\Security
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm\Security
HKEY_USERS\S-1-5-21-448539723-1644491937-725345543-
1003\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU\exe
HKEY_USERS\S-1-5-21-448539723-1644491937-725345543-
1003\Software\Microsoft\Windows\ShellNoRoam\Bags\50
HKEY_USERS\S-1-5-21-448539723-1644491937-725345543-
1003\Software\Microsoft\Windows\ShellNoRoam\Bags\50\Shell
---------------------------------
Files added:6
---------------------------------
C:\Documents and Settings\User1\Local Settings\Temp\JET2A.tmp
C:\Documents and Settings\User1\Local Settings\Temp\Perflib_Perfdata_3a0.dat
C:\WINDOWS\Prefetch\MSRLL.EXE-03966588.pf
C:\WINDOWS\Prefetch\MSRLL.EXE-107BC400.pf
C:\WINDOWS\system32\mfm\jtram.conf
C:\WINDOWS\system32\mfm\msrll.exe
---------------------------------
Files deleted:2
---------------------------------
C:\Documents and Settings\User1\Local Settings\Temp\JET29.tmp
C:\msrll.exe
```

Figure3-8

I again used the “revert to snap shot feature” in VMWare to get back to a known good state. Then I started the following applications and stopped their capture feature and cleared their logs: Filemon and TDImon. I started the capture feature on these utilities immediately before clicking on msrll.exe. Letting these programs run for about 30 seconds and stopping them, I then examined the results. Filemon supported my earlier findings of the “mfm” folder being created in the System32 directory as seen in figure 3-9, also supported were my earlier findings

that the original msrll.exe file was removed from the root of the C drive were I had placed it, as can be seen in Figure 3-10.

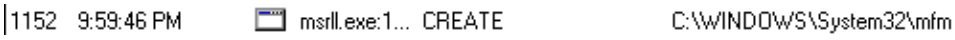

Figure3-9

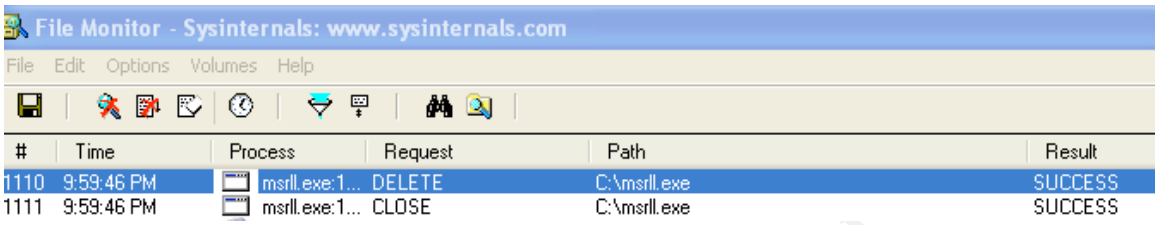

Figure3-10

TDImon showed that msrll.exe had opened port 2200 and was listening as seen in figure 3-11.

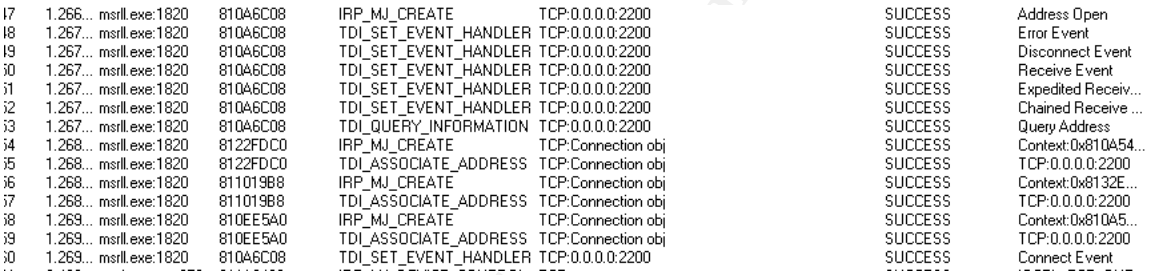

Figure 3-11

Next I looked at how msrll.exe would interact with the network. Using Netstat with the " –a" switch and TCPview, I was able to verify that the msrll.exe process was listening on port 2200 using Cavies as can be seen in figure 3-12 and in figure 3-13 using Netstat. This would be an avenue that I would later test with Telnet to see if msrll.exe would possibly accept remote commands on port 2200.

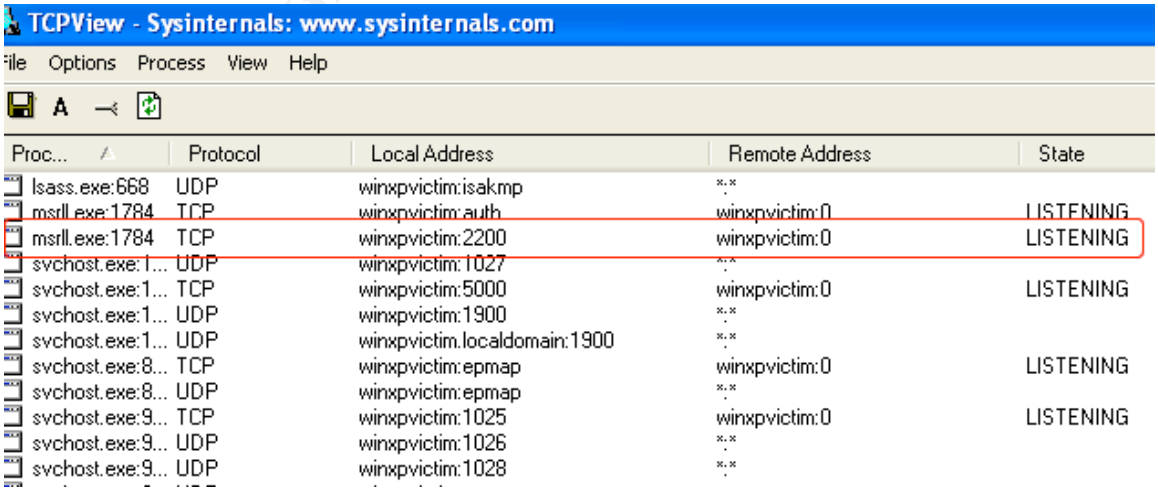

Figure 3- 12

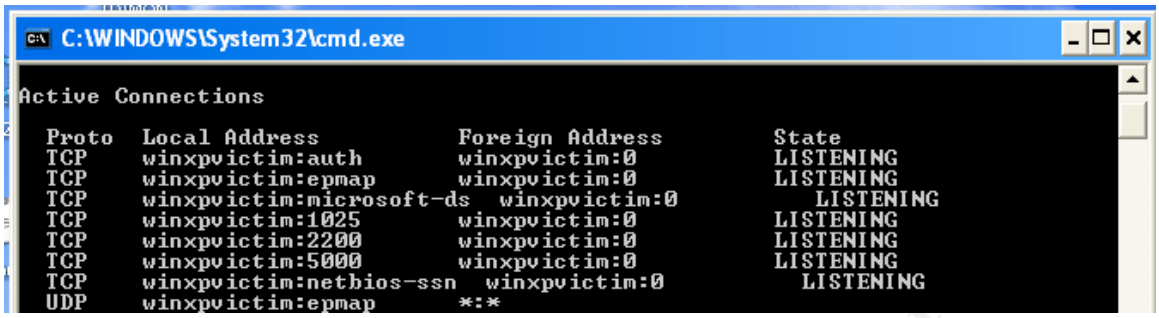

Figure3-13

Next I used my host system to telnet to the infected machine on port 2200 to try and initiate a response from the malware and was returned a "#". From there I was unable to elicit any other responses.

Using the "revert to snap shot" feature in VMWare, I set my system back to a known good state. I then used Ethereal on the host machine sniffing the VMWare virtual Ethernet adapter. I once again executed msrll.exe and I saw a dns query for collective7.zxy0.com as seen in figure 3-14.

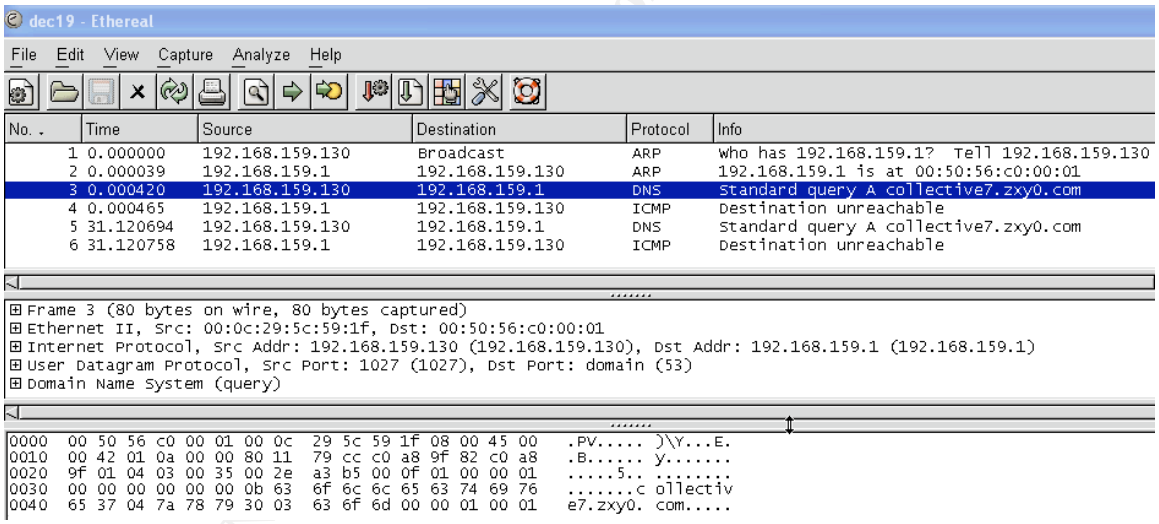

Figure3-14

Using the "revert to snap shot" feature in VMWare, I set my system back to a known good state.

As we had learned in class, in order to learn more about a malware specimen it would be necessary to mold the test environment in order to give the malware what it expects. So, next I changed the host file located on my Windows XP virtual system in the following location: C:\WINDOWS\SYSTEM32\DRIVERS\ETC, so that collective7.zxy0.com was associated with the IP address of my Linux virtual machine: 192.168.159.137 as seen in figure 3-15. I then saved this configuration as a new VMWare snapshot.

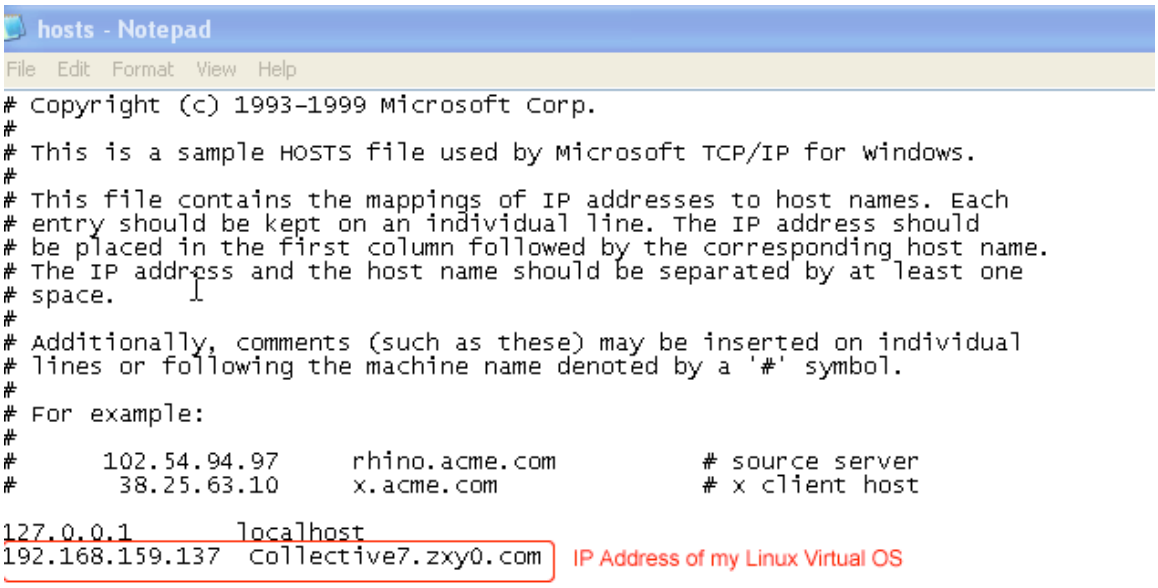

Figure3-15

After changing my host file I once again started my sniffer and restarted msrll.exe by using Windows task manager and then clicking on the msrll.exe executable in the SYSTEM32/mfm folder. I then saw failed attempts by msrll.exe to connect to ports 9999, 8080, and 6667 on collective7.zxy0.com which for now was my Linux box.

To further mold the test environment, my next step was to startup an IRC server. Using IRC-Hybrid the IRC server that was provided with the course as part of the Linux image, I started the IRC server located on my virtual Linux system by issuing the following commands:

su – ircd
./ircd
exit

ps –u ircd
irc

Then I used Ethereal on my host machine to sniff the VMWare virtual ethernet adapter for more information. I started the capture feature in Ethereal and launched msrll.exe in hopes of capturing more payloads from the packets. The sniffer logs showed that msrll.exe was making an IRC request for channel “#mils” on port 6667 as seen in figure 3-16. The IRC user names looked as though they might be random.

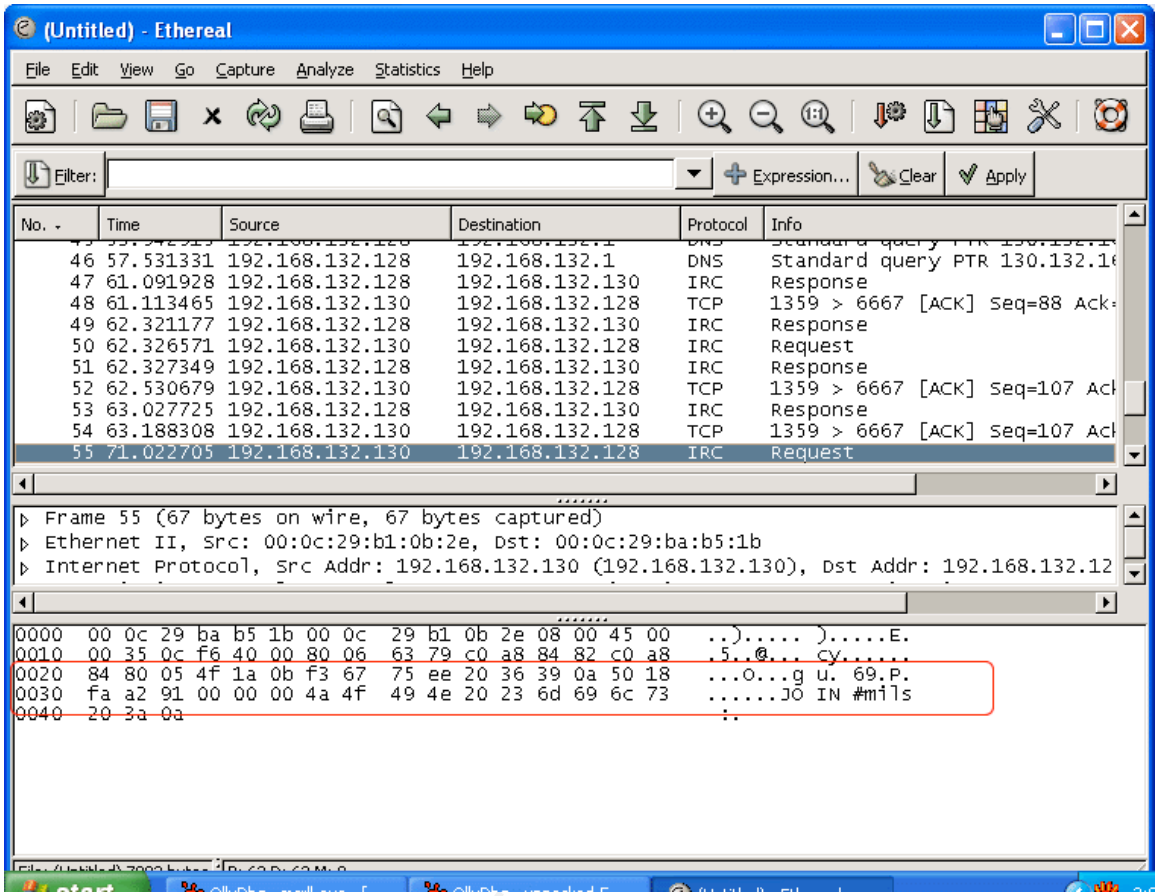

Figure 3-16

Then using my Linux box, I joined the channel #mils myself using IRC command "/join #mils".

Using the "revert to snap shot" feature in VMWare, I set my infected system back to a known good state and executed msrll.exe.

I saw a user join the channel #mils named "hIsFbHxGu". After about two minutes this connection was reset for some reason and another user joined named ZYqwcysWn. This user remained logged on during the time of my observation (approximately 15 minutes). This again confirmed that user names were possibly random.

To further test the behavior of msrll.exe, I attempted to control it from the command line. Msrll.exe could be started from the command line by typing msrll at the C:\Windows\system32\mfm prompt. From the command line I used the Taskkill utility: Taskkill /F /IM msrll.exe and was able to end the msrll.exe process. I also tried deleting the file from Windows Explorer but was denied access. I could however end the msrll.exe process from Windows Task Manager and then delete the msrll.exe file.

Section 4: Code Analysis

I began my code analysis by looking for useful strings. I knew that in order to see something useful I would need to unpack msrll.exe. To do this I would use LordPE to dump the unpacked executable from memory. I Used LordPE to locate the already running msrll.exe process in LordPE's *path pane,* I right clicked and choose "dump full" and saved the file. I then used BinText to open the file dumped from LordPE. In its unpacked form, I was able to see many interesting strings:

There were many strings that might prove useful, for the sake of brevity the following is a sampling of strings that might provide clues to the malware's functionality:

Possible Commands: ?ping, ?smurf, ?jolt, ?clones, ?clones, ?update, ?reboot, ?status, ?jump, ?nick, ?echo, ?hush, ?wget, ?join, ?akick, ?part, ?dump, ?md5p, ?free, ?update, ?hostname, ?!fif, ?play, ?copy, ?move, ?sums, ?rmdir, ?mkdir, ?exec, ?kill, ?killall, ?crash, ?sklist, ?unset, ?uattr, ?dccsk, ?killsk

Stings Indicating the Specimen Could be Possible Bot: bot.port

IRC Version in Use: mIRC v6.12 Khaled Mardam-Bey

Use of IRC Channel: irc.chan

IRC Channel: #mils

Possible Version of Malware: m220 1.0 #2730 Mar 16 11:47:38 2004

Interesting Strings to Look for Within the Code: %s bad pass from "%s"@%s, jtr.home, irc.pass, jtram.conf

Possible Web Site of Owner or Creator: collective7.zxy0.com, collective7.zxy0.com:9999! , collective7.zxy0.com:8080

Passwords Could Be Encrypted: Also listed were references to SSL and several encryption standards

The strings I saw in addition to my previous behavioral observations led me to believe that msrll.exe might be some sort of password protected bot that used IRC to issue commands.

With msrll.exe running I tried using Telnet and IRC to issue some of the commands extracted with BinText, but was unsuccessful. It was likely that msrll.exe required some sort of successful authentication before accepting commands.

My next step was to try to gain some control over msrll.exe by looking deeper into the code and hopefully finding a way to authenticate to it in order to successfully issue some of the commands found within msrll.exe's strings.

To successfully gain control of msrll.exe I would need to be able to know a password or patch msrll.exe in a way that it would not need a password. To do this I would need to pinpoint crucial instructions at the assembly code level.

I learned from the course that passwords can often be found somewhere near a "strcmp" instruction, but the results of the comparison are enforced by the "JNZ" instruction. My plan was to look for these clues and since I was not an expert at assembly code, I planned to bypass the authentication by replacing the "JNZ" instruction with the "NOP" (no operation; to do nothing) instructions using OllyDbg.

BinText provided me with some interesting strings and their associated memory positions. I could use these memory positions as starting points during the investigation of the code. I would use Ollydbg to navigate to the memory locations and look for possible ways to control msrll.exe. I knew msrll.exe was compressed using Aspack, and this would make it difficult for Ollydbg to reveal useful information. However there were a couple of options for viewing the code in an uncompressed format.

The first option was to use an unpacker to create a new uncompressed version of msrll.exe. To uncompress msrll.exe, I obtained and successfully used a tool mentioned during the course called AspackDie. To use it, I unzipped the files to my program directory. Then browsed to the AspackDie folder, clicked on the AspackDie icon and in the proceeding dialogue box I provided the path to msrll.exe, and AspackDie successfully created an unpacked version of msrll.exe.

I then used Ollydbg to open the unpacked file. I scrolled down to the memory position: 0040BB52 that was associated with "% bad pass from "%s"@%s" string found with BinText. From there I scrolled down a bit further until I was at memory location 0040BBD9. There I saw the string "dcc.pass". It looked like this might be doing something so as seen in figure 4-1, I hit the space bar and filled it with NOPs.

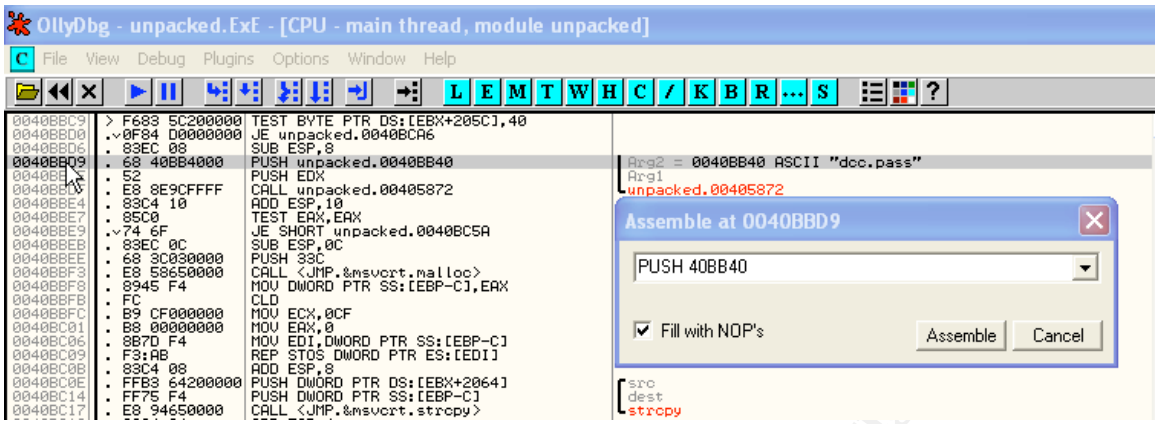

Figure4-1

When I tried to run the unpacked version of msrll.exe with Ollydbg I would get a message in the bottom right corner that the process had terminated. It is possible that the unpacking process damaged the file.

I spent a lot a time looking at the code in Ollydbg and familiarizing myself with its features. By right clicking in OllyDbg's CPU window and then choosing "search for all referenced text strings", I was able to see that near memory location 0040BB40 "dcc.pass" was a reference for "bot.port" at memory location 0040BB49 as seen in figure 4-2.

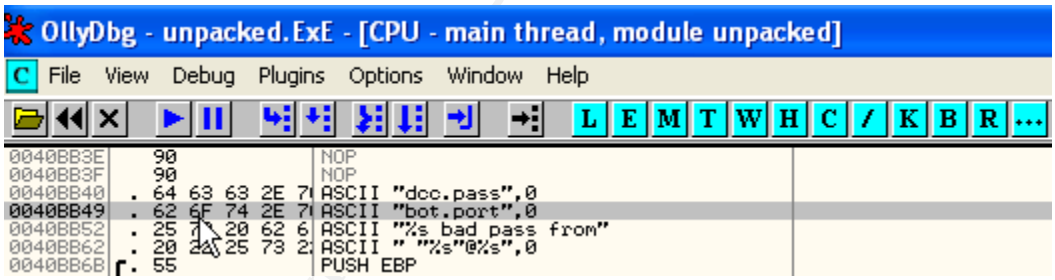

Figure4-2

I chose this location in OllyDbg and then right clicked and chose the selection for "Follow in Disassembler" as seen in figure 4-3. Then looking in the OllyDbg's dump pane, I then was able to see ASCII text mentioning port 2200 and something that looked like references to possible passwords after memory location 0040BDD9 as well as references to familiar clues found during my previous analysis.

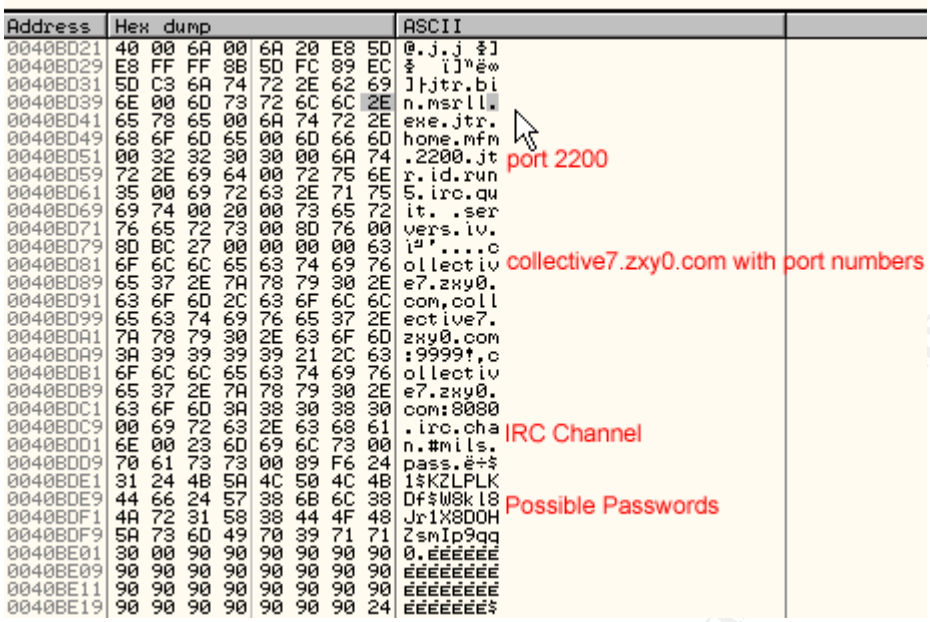

Figure4-3

Using OllyDbg, I once again right clicked and made the selection “search for” and chose “all referenced text strings”. I then saw references to “Pass” and what I had previously seen in the dump pane that looked like possible password clues at memory location 0040BDD9 as seen in Figure 4-4.

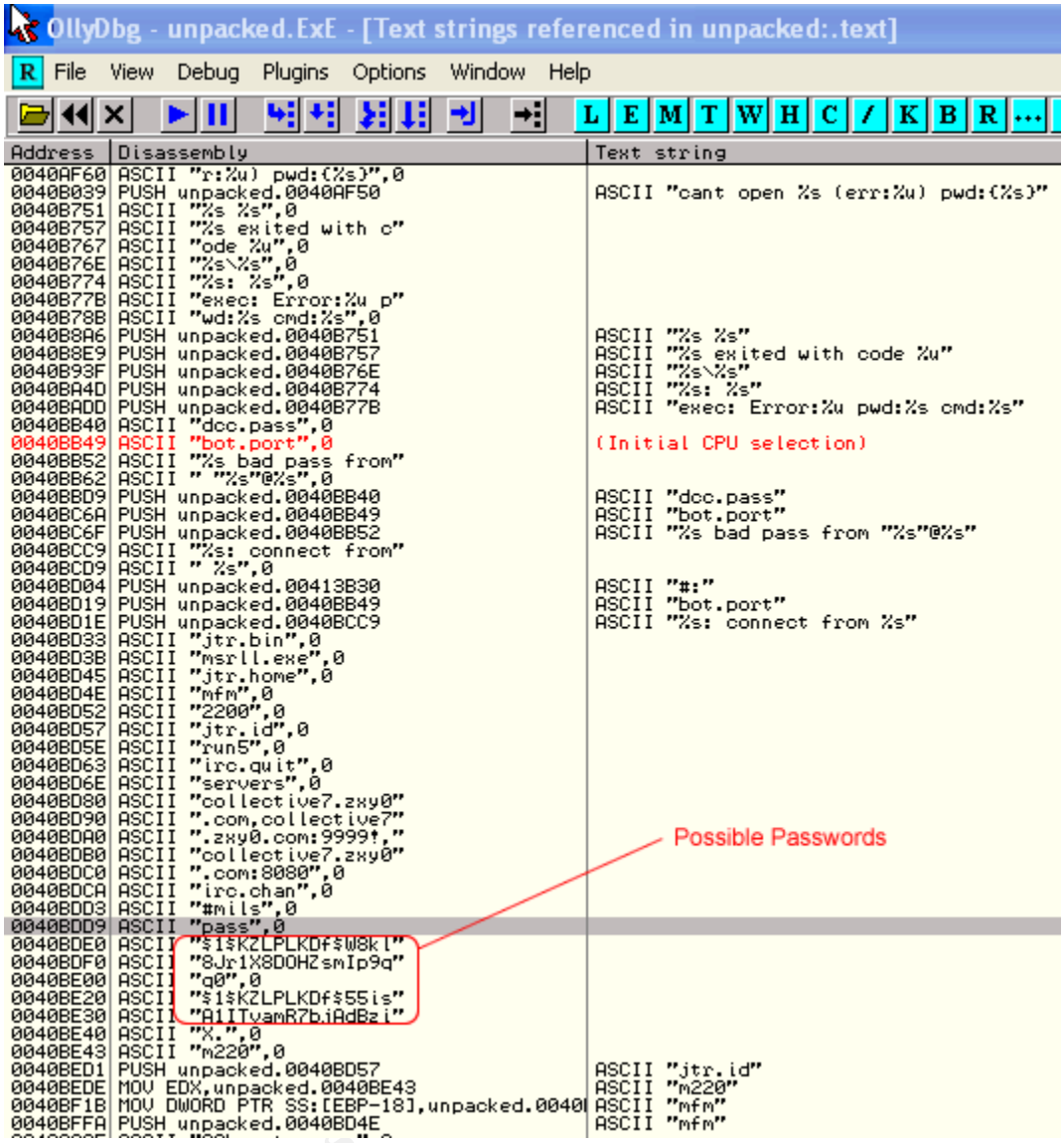

Figure4-4

I wanted to be able to set some breakpoints at key locations and try to authenticate to the malware and trigger the breakpoints. The version of msrll.exe that I unpacked using AspackDie would not run, so I needed a different way to unpack the malware executable without damaging the file.

To investigate further and produce an unpacked version of msrll.exe that would run I chose the following method. First I made sure that the msrll.exe process was not running on the virtual system. I then used Ollydbg to start the process by clicking “file open” and choosing the msrll.exe file. I then chose the run option within Ollydbg. By doing this I was avoiding dumping the file. Allowing Ollydbg to run msrll.exe let the malware unpack itself to memory. I was now beginning my

journey to look for memory position 0040BB52. I pressed “Alt+M” while in Ollydbg to get to OllyDbg’s memory map.

From there I located the first PE header. Then I chose the section that began with address 00401000. I right clicked on this location and chose “Dump in CPU”.

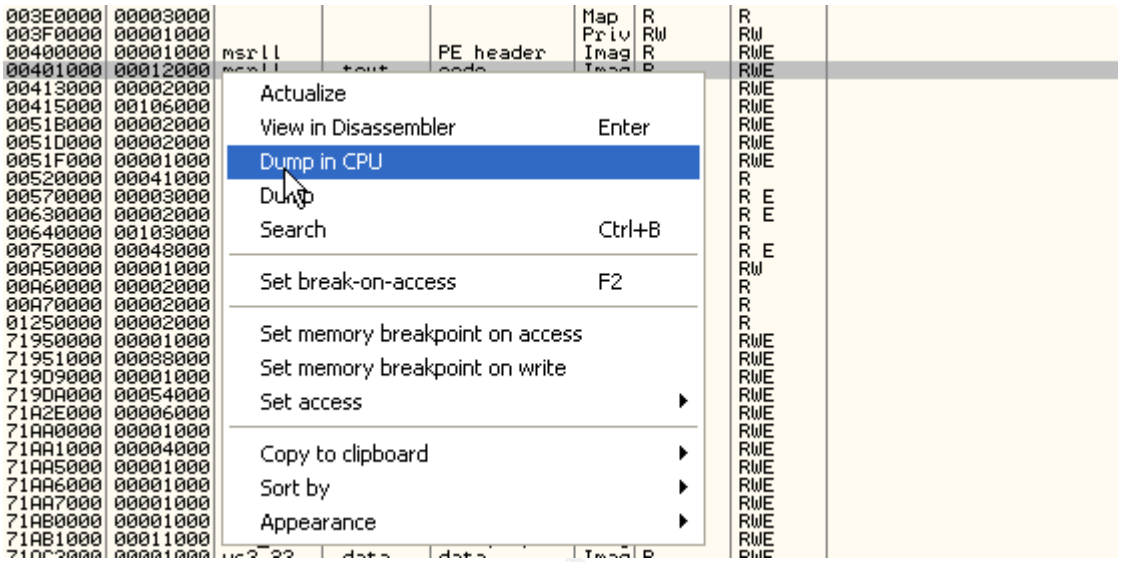

Figure4-5

Below string “dcc.pass” at memory location 0040BBE7 I found “TEST EAX, EAX”. Thinking this might be some sort of string comparison for the password I pressed the space bar and filled the instruction with “NOP” instructions hoping this would force msrll.exe to bypass a possible authentication routine. This did not work. I also tried a lot of other things that were unsuccessful. After much Trial and error, the following is what did work.

In order to get everything back to a known good state, I reverted to my snapshot. I made sure msrll.exe was not running, I then opened msrll in OllyDbg. Next I clicked on the “TEST EAX, EAX” location and pressed F2 to set a breakpoint. I then restarted msrll.exe within OllyDbg by pressing Ctrl+F2 then pressing F9 to run msrll.exe within OllyDbg. Then using Telnet from my virtual Linux installation I attempted to login to the infected Windows machine on port 2200. At the Telnet prompt, I typed the following commands found during my strings analysis: “?login” testuser" , press enter and “pass” enter. This triggered the breakpoint that I had set within OllyDbg. By looking at the Registers pane in OllyDbg I can see that the value of EAX is “00000000”. By selecting this value and right clicking and choosing “set to 1” then pressing F9 to continue running msrll.exe within OllyDbg, I was able to bypass msrll.exe’s telnet authentication. I was now able to successfully type in commands at the telnet prompt and receive responses from msrll.exe.

The following is an example of some of the commands found using BinText and their results:

?ps: listed all running processes on the infected machine
?ping: <ip> <total secs> <p size> <delay> [port]
?smurf: <ip> <p size> <duration> <delay>
?jolt: <ip> <duration> <delay>
?clone: ?clone: server[:port] amount
?clones: ?clones: [NETWORK:all] <die:join:part:raw:msg> <"parm">
?login: used to login
?uptime: shows uptime of the system and uptime of the bot
?reboot: reboots the infected system
?status: shows yes or no for service, user, inet connection, contype, reboot privs
?nick: set an irc sock to perform ?nick command on
?hush: set an irc sock to perform ?hush command on
?join: set an irc sock to perform ?join command on
?md5p: <pass> <salt>
?free: ?free <cmd>
?update: <url> <id>
?hostname: host name of infected system
?play: (null) somefile
?sums: sums of files located within WINDOWS/SYSTEM32/mfm directory
?mkdir: lets you create a directory in the WINDOWS/SYSTEM32/mfm directory
?rmdir: lets you remove a directory in the WINDOWS/SYSTEM32/mfm directory
?exec: executes the program specified to run in the background
?kill: when given the process id, ends the process

I spent quit a bit of time trying to achieve control of msrll.exe via IRC, but was not successful. It is possible that via IRC I was not sending commands in the format that the malware would accept. But during this time, one of the ways that I learned to use OllyDbg to investigate the malware specimen was to right click and search for all referenced text strings. Once there, I right click and chose "Set Breakpoint on Every Command". Then press F9 to run the executable within OllyDbg. You will hit a lot of break points, but this slow motion view of the executable can be very informative. When you break in an area that is not of interest press F9 to continue running, if you break in an area that is of interest press F7 to single step through the code.

Section 5: Analysis Wrap-up

Section 5.1 Summary of Msrll.exe

Msrll.exe exhibits behaviors that are often associated with malicious bots. Msrll.exe is designed to start each time Windows is started and connect to an IRC channel. Although this particular malware does not attempt to disguise itself, as its process can be easily seen in Windows XP's Task Manager, most users would not notice its presence. Msrll.exe listens and accepts commands on port

2200 and connects to IRC channel #mils on port 6667. It appears that commands can be sent to msrll.exe via IRC on channel #mils and Telnet on port 2200 by the proper use of a pass key. Commands found within the code of msrll.exe such as ping, smurf, crash, mkdir, etc. are indicative of DDos and hacker type behaviors such as illegal file storage or "owning" someone else's computer.

Based on my analysis, msrll.exe strongly resembles an IRC bot sometimes referred to as a zombie. Results found with BinText and OllyDbg point to commands that would allow the bot owner to control unsuspecting users Windows based system with msrll.exe. IRC provides a way for the hacker to control msrll.exe in a practically anonymous fashion. It would be relatively easy for a hacker to control a compromised computer or computers without being detected. By using Msrll.exe the hacker would not need to scan for open ports because msrll.exe will start during system startup, joining an IRC channel and await commands. By having msrll.exe installed on many computers a person with bad intentions would have the ability to send commands to many computers simultaneously via IRC.

Individuals interested in "owning" systems in such a way as to have a bot army awaiting commands via IRC, could include anybody with motive or desire to illegally deny service to an organizations web presence or disrupt network services through DDos attacks.

Section 5.2 Additional Findings

My next note brings up an important point regarding malware analysis. It is important to make notes when things happen. My example comes from the fact that I used my browser to go to the web address "collective7.zxy0.com" when it was identified as a string using BinText. At the time I saw a web page there that looked as though it was offering some type of email service, but unfortunately I didn't make notes or a screenshot and on later visits I was not able to pull up a page. I took the first part "collective7" of the web address off and used my browser to go to this website address: zxy0.com, a derivative of the site "collective7.zxy0.com". I saw a site that wasn't necessarily incriminating, but intriguing. The site made references to things such as: "you can't get this shit in stores" and "feel free to paypal me if you like my software".

I looked at the source code for the webpage and under the meta tags I saw "home of m220-Beetlework". Possibly by coincidence one of the strings found within the code was: m220 1.0 #2730 Mar 16 11:47:38 2004

Interestingly by typing in m220.exe into the Google search engine, I find some results at the TrendMicro anti-virus site. TrendMicro discusses a backdoor with two executables an ftp server DTRAN.exe and a malicious IRC bot M220.exe. The descriptions given by TrendMicro seem to be similar to what I have found with msrll.exe. Msrll.exe is mentioned in the TrendMicro article when TrendMicro identifies the creation of the mfm folder containing the msrll.exe file.

In order to get the most out of my learning I did not do a web search for msrll.exe or run antivirus software against it. But at the end of my analysis when I did run a web search, it is interesting that a derivative of the web address that I found the msrll.exe using had an owner by the nickname m220-Beetlework and TrendMicro had an associated article involving the keyword msrll.exe and a malicious IRC bot labeled M220.exe.

Section 5.3 Detecting Msrll.exe

Running updated anti-virus software and a fully patched system will go a long way in keeping malicious programs such as msrll.exe off of your computer. It is also useful to be aware of and monitor the processes that normally run on your system. Having a personal firewall could be useful in helping monitor incoming and outgoing traffic generated by malware like msrll.exe. Network administrators should become familiar with IRC's malicious uses and ports used due to its common association with malware.

It is quite possible since communication is established from within, that your firewall will not help in protecting against IRC bots and antivirus software may not provide adequate protection due to the fact that malware writers can stay one step ahead of antivirus companies and create versions of malware that are not recognized by current antivirus signatures.

The Netstat utility can be helpful in identifying IRC bots. Since the most common IRC channels use port 6666 and 6667, and IRC often uses the Ident protocol running on port 113. Netstat can be used in the following way to help identify the use of these ports on a local system. At the command line type the following commands and press enter:

netstat –an | find ":6667"

If you don't have your own IRC session running it is possible that malware is trying to compromise your system via IRC if you see results similar to the following:

TCP 192.168.159.130:1033 192.168.159.137:6667 ESTABLISHED

To check for the Ident protocol use: nestat –an | find ":113"

Other options might include network professionals monitoring for IRC traffic. In our particular case with msrll.exe network professionals could monitor network traffic for packets containing #mils.

Section 5.4 Removal of msrll.exe

Removal of msrll.exe seemed to be relatively simple on my Windows XP system. First you will need to check and see if the msrll.exe process is running. If it is running it can be stopped by pressing CTRL + ALT + DEL. Then you will be able

to browse to the “mfm” folder location and delete the folder. This essentially stopped msrll.exe from running as it appears to have no failover mechanism. There was still a registry entry that you might like to clean up. You can do this by clicking on Start, then Run and typing Regedit and enter. This will bring up the registry editor. Look for this path: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\mfm and delete the entry. Once completed restart your system and check to make sure your removal attempts were successful.

References

Aspack. Vers. 1.32 Dec. 2004 <http://www.aspack.com/>

AspackDie. Vers. 1.41 Nov. 2002 <http://scifi.pages.at/yoda9k/>

AutoRuns. Vers. 6.1 Dec. 2004
http://www.sysinternals.com/ntw2k/freeware/autoruns.shtml

BinText. Vers. 3.00 Nov. 2000
<http://www.foundstone.com/index.htm?subnav=resources/navigation.htm&subcontent=/resources/proddesc/bintext.htm>

Ethereal. Vers. 0.10.8 Dec. 2004 <http://www.ethereal.com>

Filemon. Vers. 6.12 Oct. 2004
<http://www.sysinternals.com/ntw2k/source/Filemon.shtml>

InstallWatch Pro. Vers. 2.5 May 2000
<http://www.epsilonsquared.com/installwatch.htm>

IRCd-Hybrid. Vers. 2.8/hybrid-6.3.1 June 2002 <http://www.ircd-hybrid.net>

MD5sum. Nov. 1999 <http://www.gnu.org/software/textutils/textutils.html>

LordPE. Vers. 1.31 March 2002
<http://mitglied.lycos.de/yoda2k/LordPE/info.htm>

Ollydbg. Vers. 1.0.10.0 May 2004 <http://home.t-online.de/home/Ollydbg>

Process Explorer. Vers. 8.61 Dec 2004
<http://www.sysinternals.com/ntw2k/freeware/procexp.shtml>

Regmon. Vers. 6.12 Aug. 2004
<http://www.sysinternals.com/ntw2k/source/Regmon.shtml>

Regshot. Vers. 1.61e5 Jan. 2003
<http://www.pcworld.com/downloads/file_description/0,fid,19540,00.asp>

TDImon. Vers. 1.01 July 2000
<http://www.sysinternals.com/ntw2k/freeware/TDImon.shtml>

VMWare Workstation. Vers. 4.5.1 March 2004 <http://www.vmware.com>

Last Updated: October 18th, 2017

Upcoming Training

SANS Paris November 2017	Paris, France	Nov 13, 2017 - Nov 18, 2017	Live Event
Community SANS Ottawa FOR610	Ottawa, ON	Dec 04, 2017 - Dec 09, 2017	Community SANS
SANS Cyber Defense Initiative 2017	Washington, DC	Dec 12, 2017 - Dec 19, 2017	Live Event
SANS Security East 2018	New Orleans, LA	Jan 08, 2018 - Jan 13, 2018	Live Event
SANS vLive - FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques	FOR610 - 201801,	Jan 22, 2018 - Feb 28, 2018	vLive
SANS Dubai 2018	Dubai, United Arab Emirates	Jan 27, 2018 - Feb 01, 2018	Live Event
Cyber Threat Intelligence Summit & Training 2018	Bethesda, MD	Jan 29, 2018 - Feb 05, 2018	Live Event
SANS London March 2018	London, United Kingdom	Mar 05, 2018 - Mar 10, 2018	Live Event
SANS Secure Singapore 2018	Singapore, Singapore	Mar 12, 2018 - Mar 24, 2018	Live Event
SANS Secure Canberra 2018	Canberra, Australia	Mar 19, 2018 - Mar 24, 2018	Live Event
SANS Munich March 2018	Munich, Germany	Mar 19, 2018 - Mar 24, 2018	Live Event
Community SANS Columbia FOR610	Columbia, MD	Mar 26, 2018 - Mar 31, 2018	Community SANS
SANS 2018	Orlando, FL	Apr 03, 2018 - Apr 10, 2018	Live Event
SANS Security West 2018	San Diego, CA	May 11, 2018 - May 18, 2018	Live Event
DFIR Summit & Training 2018	Austin, TX	Jun 07, 2018 - Jun 14, 2018	Live Event
Community SANS Columbia FOR610	Columbia, MD	Aug 20, 2018 - Aug 25, 2018	Community SANS
SANS OnDemand	Online	Anytime	Self Paced
SANS SelfStudy	Books & MP3s Only	Anytime	Self Paced